ЛАБОРАТОРНАЯ РАБОТА № 1

КЛАССЫ

1.1. Постановка задачи

Цель настоящей работы состоит в ознакомлении студента с правилами организации классов, принятыми при программировании на языке C++. В процессе выполнения настоящей работы каждый студент должен разработать два класса и написать тестовые программы для демонстрации их работоспособности.

1.2. Варианты заданий

Номер	Задачи
1	1 и 14
2	2 и 15
3	3 и 16
4	4 и 14
5	5 и 15
6	6 и 16
7	7 и 14
8	8 и 14
9	9 и 15
10	10 и 16
11	11 и 14
12	12 и 15
13	1 и 16
14	2 и 14
15	3 и 15
16	4 и 16
17	5 и 14
18	6 и 15
19	7 и 16
20	8 и 14
21	9 и 15
22	10 и 16
23	11 и 14
24	12 и 15

1.3. Задачи

Ниже приводятся формулировки задач, которые предлагаются студентам для решения в настоящей лабораторной работе.

Предлагаемые задачи сгруппированы по тематике. В разделе принята сквозная нумерация задач. В каждом из разрабатываемых классов должен быть предусмотрены, по крайней мере, три конструктора: конструктор умолчания, конструктор с параметрами, конструктор копирования. Для классов, в которых конструктор выделяет память, должен быть предусмотрен деструктор. В классах, используемых для хранения, должны быть предусмотрены методы доступа.

1. 3. 1. Математический вектор

В математике широко используется понятие вектора. В этом разделе приводятся формулировки двух задач, которые отличаются требованиями к реализации разрабатываемого вектора. Над объектами разрабатываемого класса в каждой из этих задач должны быть предусмотрены следующие основные операции:

- сложение векторов,
- вычитание векторов,
- вычисление скалярного произведения векторов,
- вычисление длины вектора.

Задача 1.

Математический вектор реализуется на основе статического массива. Тип элементов, хранящихся в массиве, выбирается студентом самостоятельно. Класс должен содержать поле для хранения размерности вектора.

Задача 2.

Математический вектор реализуется на основе динамического массива. Тип элементов, хранящихся в массиве, выбирается студентом самостоятельно. Класс должен содержать поле для хранения размерности вектора.

1. 3. 2. Строки

Задачи 3 - 6.

Необходимость работы со строками возникает при решении многих задач. Ниже предлагаются четыре задачи, связанные с разработкой класса `String`, предназначенного для работы со строками (задачи 3 - 6). Задачи различаются наборами операций, которые можно выполнять со строками и требованиями которые предъявляются к их реализации.

Задача 3.

Необходимо предусмотреть выполнение следующих операций над строками разрабатываемого класса:

- **int** Length() – определение длины строки,
- **void** Copy(**const** String& str) – скопировать строку str,
- **int** Find(**char** ch, **int** start) – начиная с индекса start, найти положение символа ch в строке, для которой вызывается метод Find(),
- int FindLast(**char** ch) – найти последнее вхождение символа **ch**,
- String Substr(**int** index, **int** count) – выделение подстроки, начиная с индекса index; count – длина, выделяемой подстроки,
- **void** Remove(**int** index, **int** count) – удаление подстроки, начиная с индекса index; count – длина удаляемой подстроки.
- **void** Insert(**char*** s, **int** index) - вставка строки в стиле языка С в строку, для которой вызывается метод Insert; index – позиция, перед которой выполняется вставка.
- **void** print() – метод для вывода строки на экран дисплея.

Требования к реализации. Строка должна располагаться в динамической памяти. Необходимо предусмотреть поле для хранения информации о длине строки.

Задача 4.

Необходимо предусмотреть выполнение следующих операций над строками разрабатываемого класса:

- **int** Length() – определение длины строки,
- void Copy(const String str) – скопировать строку str,
- int Find(char ch, int start) – начиная с индекса start, найти положение символа ch в строке, для которой вызывается метод Find(),
- int FindLast(char ch – найти последнее вхождение символа ch,
- String Substr(int index, int count) – выделение подстроки, начиная с индекса index; count – длина, выделяемой подстроки,
- void Remove(int index, int count) – удаление подстроки, начиная с индекса index; count – длина удаляемой подстроки.
- void Insert(char* s, int index) - вставка строки в стиле языка С в строку, для которой вызывается метод Insert; index – позиция, перед которой выполняется вставка.
- void print() – метод для вывода строки на экран дисплея.

Требования к реализации. Строка должна располагаться в динамической памяти. Поле для хранения информации в классе отсутствует.

Задача 5.

Необходимо предусмотреть выполнение следующих операций над строками разрабатываемого класса:

- int Length() – определение длины строки,
- void Copy(const String& str) – скопировать строку str,
- void Copy(const char* str) – скопировать строку str,
- String Substr(int index, int count) – выделение подстроки, начиная с индекса index; count – длина, выделяемой подстроки,
- void Remove(int index, int count) – удаление подстроки, начиная с индекса index; count – длина удаляемой подстроки.
- void Insert(char* s, int index) - вставка строки в стиле языка С в строку, для которой вызывается метод Insert; index – позиция, перед которой выполняется вставка.
- void trim() – удалить из строки ведущие и завершающие пробелы,
- void read() – ввести строку с клавиатуры,
- void print() – метод для вывода строки на экран дисплея.

Требования к реализации. Строка должна располагаться в динамической памяти. Необходимо предусмотреть поле для хранения информации о длине строки.

Задача 6.

Необходимо предусмотреть выполнение следующих операций над строками разрабатываемого класса:

- int Length() – определение длины строки,
- void Copy(const String& str) – скопировать строку str,
- void Copy(const char* str) – скопировать строку str,
- String Substr(int index, int count) – выделение подстроки, начиная с индекса index; count – длина, выделяемой подстроки,
- void Remove(int index, int count) – удаление подстроки, начиная с индекса index; count – длина удаляемой подстроки.
- void Insert(char* s, int index) - вставка строки в стиле языка С в строку, для которой вызывается метод Insert; index – позиция, перед которой выполняется вставка.
- void trim() – удалить из строки ведущие и завершающие пробелы,
- void read() – ввести строку с клавиатуры,

- void print() – метод для вывода строки на экран дисплея.

Требования к реализации. Строка должна располагаться в динамической памяти. Поле для хранения информации в классе отсутствует.

1. 3. 3. Одномерный массив

Задачи 7 – 10.

В задачах настоящего раздела необходимо разработать класс, который может рассматриваться как улучшенный одномерный массив. Одним из “улучшений”, которые должны быть предусмотрены в разрабатываемых классах, является наличие так называемого “счетного” режима. При работе в счетном режиме производится подсчет количества элементов, которые хранятся в массиве. Для реализации такого режима в классе следует предусмотреть специальное поле.

Реализация такого класса предусматривает использование динамического массива. В задачах № 7 и 9 следует создать одномерный массив, который можно назвать условно массивом в стиле языка Си. Конструктор такого массива должен задавать количество элементов, которые будут храниться в массиве. Минимальное значение индекса равно нулю

В задачах № 10 и 12 следует создать одномерный массив, который можно назвать условно массивом в стиле языка Паскаль. Конструктор такого массива должен задавать минимальное и максимальное значения индексов.

Задачи различаются наборами операций, которые можно выполнять с массивами и требованиями, которые предъявляются к их реализации. Ниже приводятся наборы операций для каждой задачи.

Задача 7 Разрабатываемый класс должен предусматривать выполнение следующих операций:

- копирование массивов,
- ввод элементов массива с клавиатуры,
- вывод элементов массива на экран дисплея,
- добавка элемента в конец занятой части массива,
- объединение двух массивов,
- вычисление суммы элементов массива.

Задача 8 Разрабатываемый класс должен предусматривать выполнение следующих операций:

- копирование массивов,
- ввод элементов массива с клавиатуры,

- вывод элементов массива на экран дисплея,
- добавка элемента в конец занятой части массива,
- удаление части массивов; удаляемые элементы задаются значением индекса, с которого необходимо начать удаление и количеством удаляемых элементов,
- вычисление произведения элементов массива.

Задача 9 Разрабатываемый класс должен предусматривать выполнение следующих операций:

- копирование массивов,
- ввод элементов массива с клавиатуры,
- вывод элементов массива на экран дисплея,
- добавка элемента в конец занятой части массива,
- вставка одного массива в заданную позицию другого,
- вычисление значения наибольшего элемента массива.

Задача 10 Разрабатываемый класс должен предусматривать выполнение следующих операций:

- копирование массивов,
- ввод элементов массива с клавиатуры,
- вывод элементов массива на экран дисплея,
- добавка элемента в конец занятой части массива,
- умножение всех элементов массива на заданное число,
- вычисление разности между максимальным и минимальным значениями элементов массива.
- вычисление произведения элементов массива.

1. 3. 4. Матрица

Задачи 11 и 12

Необходимо разработать класс, инкапсулирующий динамическую матрицу. Память для такой матрицы должна выделяться во время выполнения программы. В задаче № 11 следует создать матрицу, которую можно условно назвать матрицей в стиле языка Си. Конструктор такой матрицы должен определять количество строк и столбцов. Минимальное значение индекса для строк и столбцов должно быть принято равным нулю. В задаче № 12 следует создать матрицу, которую можно условно назвать матрицей в стиле языка Паскаль. Конструктор такой матрицы должен определять минимальное и максимальное значения индекса для строк и столбцов.

Разрабатываемые классы должны обеспечить выполнения следующих операций:

- сложение матриц,

- вычитание матриц,
- умножение матриц (факультативно),
- вывод матрицы на экран дисплея,
- ввод элементов матрицы с клавиатуры,
- вычисление суммы элементов каждой строки,
- вычисление суммы элементов каждого столбца.

1. 3. 5. Полином

Задача 13.

Под полиномом понимается алгебраическое выражение следующего вида:

- $a_n x^n + a_{n-1} + ..+ a_1 + a_0$.

Разработать класс, моделирующий математическое понятие полинома. Такой класс может быть построен либо на основе динамического массива, либо на основе связанного списка, Студент может самостоятельно выбрать вид реализации. Класс должен обеспечивать выполнение следующих операций:

- вычисление значения полинома для заданного значения аргумента,
- сложение двух полиномов,
- вывод полинома на экран дисплея.

1. 3. 6. Рациональные числа

Задача 14.

Рациональные числа – это множество частных вида P / Q, где P и Q – целые числа, причем Q != 0. Число P называется числителем, а Q – знаменателем.

Такое представление чисел не является однозначным. Например,

2 / 5 == 4 / 10 == 12 / 30.

Обычно результат вычислений с рациональными числами приводят к так называемой редуцированной форме, когда числитель и знаменатель не имеют общего знаменателя. Чтобы выполнить преобразование рационального числа к редуцированной форме, числитель и знаменатель необходимо разделить на их наибольший делитель(GCD, greatest common denominator).

Разработанный класс(Rational) должен обеспечить выполнение следующих операций:

- конструктор умолчания.
- конструктор с параметрами.
- сложение,
- вычитание,

- умножение,
- деление,
- сравнения на равенство и неравенство.
- отношения.
- вывод рационального числа на экран дисплея.

Замечание. После выполнения арифметических операций результат должен быть преобразован к редуцированной форме.

1. 3. 7 Комплексные числа

Задачи 15 - 16

В этих задачах требуется разработать класс, обеспечивающий работу с комплексными числами. Задачи отличаются своей реализацией. В задаче 15 реализация должна содержать два поля, определяющие соответственно действительную и мнимую часть комплексного числа. В задаче 16 реализация должна содержать три поля. Первое и втрое поле должны задавать само комплексное число (его действительную и мнимую части), а последнее поле должно содержать модуль комплексного числа.

Разработанный класс(Complex) должен обеспечить выполнение следующих операций:

- сложение,
- вычитание,
- умножение,
- деление,
- вывод комплексного числа на экран дисплея.

1. 3. 8. Числа с фиксированной точкой

Задача 17

В этой задаче требуется разработать класс, работающий с действительными числами. Класс должен использовать представление чисел в форме с фиксированной точкой.

Математически число с фиксированной точкой можно рассматривать как частный случай чисел с плавающей точкой. Действительно, числа с плавающей точкой задаются следующим образом

$m * b^e$, где m – мантисса, b – основание системы счисления, а e – порядок.

Предположим теперь, что e – постоянная величина, определяющая количество разрядов в дробной части числа. В этом случае переходим к представлению числа в форме с фиксированной точкой.

Класс, предназначенный для реализации чисел с фиксированной точкой должен содержать, по крайней мере, два поля. Одно для представления мантиссы (m), а второе для

представления порядка (e). Поле m должно быть индивидуальным для каждого объекта (числа), а второе поле (e) одинаковым для некоторого множества чисел. С учетом этого поле e должно быть объявлено статическим. Оба поля могут иметь тип int.

Класс должен обеспечивать выполнение всех операций, которые характерны для действительных чисел.